JIM V1: packaging and installation strategy (DRAFT)

Gabriele Garzoglio,
Feb 5, 2003
Apr 10, 2003

Abstract

This document highlights the differences on the packaging and installation strategy between JIM v0 and v1 and reports design decisions. It is a technical document, which relies on the knowledge of the organization of the packages of JIM V0.

Use of XML

During the development of JIM V0, we explored the management of the configurations of various JIM products using XML. This approach turned out to be flexible and extensible and we decided to base the whole configuration management of V1 in XML. Configuration files in this context are files that either configure the behavior of the jim products or describe the configuration of resources at a site.
A set of new tools has been developed to allow XML manipulation of configuration files. Central to the release of V1 is the use of the “xml_meta_configurator”, a tool used to create configuration files in XML. We have also introduced the use of “galax”, an implementation of Xquery fully compliant with the W3C specifications, to manipulate the configuration files. As in V0, python scripts use “pyxml” to parse and manipulate XML documents.

Installation and configuration strategy

This paragraph is the result of a discussion between Tom Rockwell, Igor Terekhov and Gabriele Garzoglio. Tom is going to work on this task.
The installation procedure for JIM V0 is well understood and debugged, but it is complex. The outcome of this work should lead to an easy installation procedure for JIM V1.
We have decided to brake down the installation of the JIM suite into 2 distinct phases: product installation and product configuration. This is somewhat different from the current installation procedure, where each product is installed and configured separately.
In V1, there is an “umbrella” product, “samgrid”, whose responsibility is installing the whole JIM V1 suite, according to the specifications of the administrators. “samgrid” can be packaged as a ups product and rely on ups for product installation.
The “samgrid” installation action will

1. Interact with the installer in order to create an installation plan. The interaction is driven by the “xml_meta_configurator” and the plan is saved in XML. Some examples of the questions to the installer are “Do you want to install the samgrid client, the submission and queuing system, the broker infrastructure, the execution framework, the monitoring framework?”, “Where is the local area where to install the software (need root write access) ?”

2. Gather information that may be necessary upfront in order to install some products, before even tailoring them. Currently some jim products need this interaction in order to be installed; we need to investigate if these step can be deferred until tailoring of the products.
3. Interact with the central product server (today ups/upd at FNAL) in order to install the products needed. This step should provide the transfer of the packages to the remote site in a reliable way e.g. by automatically retrying the single products installation commands (namely "upd install")

The "samgrid" configuration (tailor) action will

1. Interact with the installer in order to produce a site configuration document, when applicable i.e. depending on the installation plan. For example, when installing an execution site, samgrid needs to record the description of the resources; this is not needed when installing the client software or the queuing system or the brokering infrastructure.
2. Include in the site configuration the knowledge of the grid global parameters. For example, the location of the broker or the default log server (central for now). This knowledge is currently kept in the product "jim_config", which we don't use anymore. This knowledge will be part of the environment to run servers (responsibility of "server_run") and we'll be setup when using the client (we need to develop the tool for extracting this information from the site configuration and make it available as environment variables).
3. Run the tailoring procedure of each product in turn. In principle, since each product uses the xml_meta_configurator for the tailoring procedure and samgrid known where each of the products are, the questions to configure all the products can be gathered all upfront. In practice, we should investigate whether all the products can be configured even if their dependencies have not been configured yet

Details

1. Each product will have its own configuration file. The site configuration should express what packages of the JIM suite are installed and provide pointers to the local configuration files. This will be useful for monitoring purposes.
2. For the initial version, samgrid will be available via ups/upd. We are considering of providing a bootstrap script that installs ups/upd; this would be useful for those configurations where sam is not needed, namely "client", "submission", "brokering" sites.
3. Each product that samgrid can install will have 3 ups actions:
   i. ask: this action starts up the interview with the administrator to gather the configuration information needed for the given product; it saves the configuration in the etc area of the product
   ii. act: this action accepts as an input the XML document of the "answers" created above and takes the necessary actions to implement the product configuration (e.g. create directories, configuration files, etc.)
   iii. tailor: this action calls the "ask" then the "act" actions

4. Product configuration and meta-configuration (which drives xml_meta_configurator through the interview) are both located in the etc area of the product
5. Some configuration steps (“act” action) can be taken only as root. Each product must check if it has enough privileges to take certain actions; if not exit with an error code (for now “exit 2”)
6. Each interview meta-configuration file will have an xml tag called “interview_schema_version” to introduce a versioning scheme of the form x_x. This ordering facilitates product upgrade, since for a new version of a product it may still be possible to use old answers to be configured
7. Version consistency among the products of the JIM suite is managed tagging “current” (“old”, “test”, etc.) the products in upd. Updates at a site can be automated by analyzing what’s current via upd versus what’s current locally.
8. samgrid will not change the content of any interview files
9. xml_meta_configurator should be upgraded to accept (as 3rd argument) an XML file of default answers; defaults provided via this mechanism should have higher precedence to the default mechanism specified in the interview XML file; the XML default file should be able to specify whether it is needed to ask the question indicating the default, or accepting the default as an answer silently. This mechanism would facilitate the communication of common set of parameters between samgrid and the various products; it would also facilitate product upgrades (see also point 6)

The packaging strategy

The dependencies of the packages of the JIM V1 software suite are shown in Figure 1.

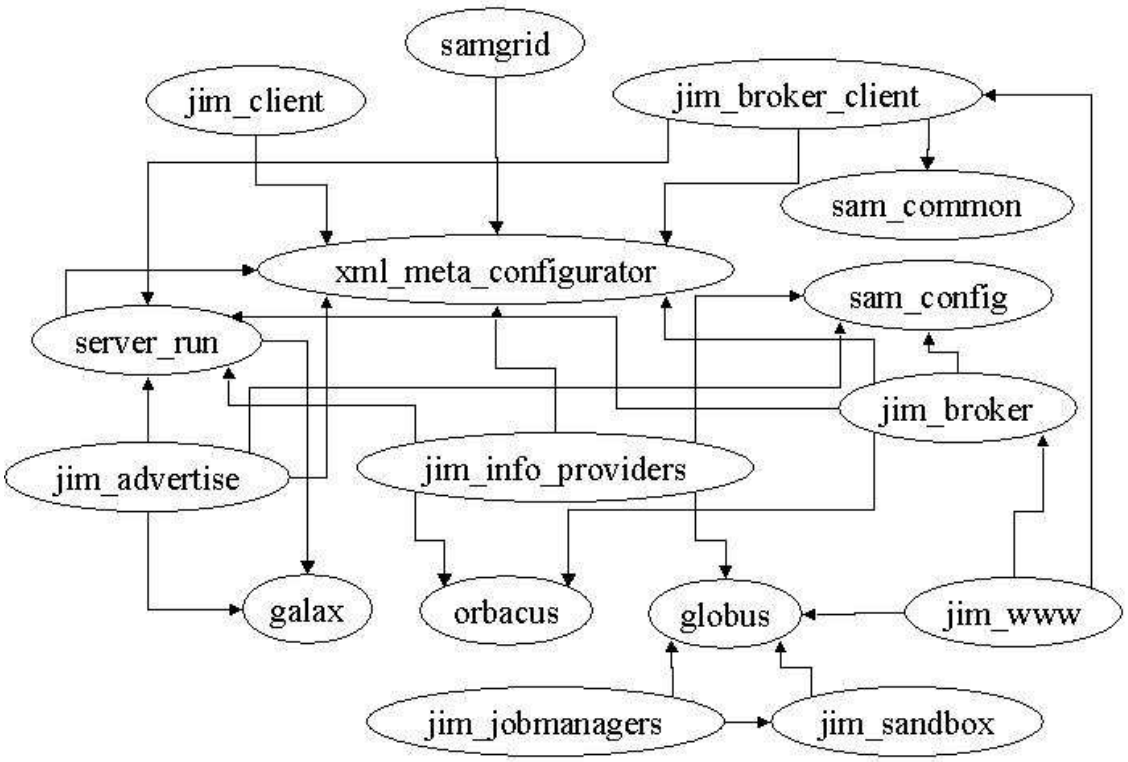

Figure 1: The products of the JIM V1 software suite. Dependencies are indicated by arrows. For example, "samgrid" depends on "xml_meta_configurator".

There are a few novelties with respect to JIM V0.

1. the central role of the xml_meta_configurator; all the jim products that need deployment depend on it
2. the introduction of "galax". Galax is an implementation of Xquery fully compliant with the W3C specifications
3. the introduction of the "jim_client" package and the new role of "samgrid". JIM V1 adopts a 3-tier architecture implemented in Condor-G v6.5: submission client and queuing system are here physically and conceptually separated. The samg client (SAMGrid-JDL compiler + condor_submit infrastructure), currently in the package "samgrid", will be moved to the new jim_client; the job queuing mechanism will stay in "jim_broker_client"; samgrid will become a pure "umbrella" product to manage the installation of the jim suite (see "Installation and configuration strategy")
4. the product "jim_info_providers" of JIM V0 is split in two: resource advertising mechanism is moved to "jim_adverise"; configuration of globus MDS + implementation of the MDS information providers stays in "jim_info_providers". This separation is somewhat natural: distinct people maintain the two parts; "jim_info_providers" V1 is flavor dependent, while jim_advertise is not (it'sall python code); the two products were initially put together with the goal of creating single information providers, that could be used by both MDS and

jim_advertise: this hasn't happened during V0 and will not happen in V1; if this becomes an issue for V2, we can rearrange the organization of the two products.

5. the product "jim_config" does not exist anymore. Global parameters (such as the location of the broker) will be defined in the environment by looking at the site configuration file. "server_run" can already do this for the jim servers; we need to develop the tool for the clients (the tool could then be wrapped in a ups table file for homogeneity).
6. we should try to minimize the dependency of sam_config to samgrid by saving relevant sam_config information (e.g. sam corba naming servers IORs) to the site configuration only (Figure 1 shows all the products that need information from sam_config)
7. the product jim_jobmanegers contains the experiment specific submission/monitoring/cleaning commands to the underlying analysis facilities (e.g. sam, caf, d0mc); the entry points to these facilities (e.g. the location of the sam command) will be available to the job managers as globus job manager tools (libexec/globus-gram-job-manager-tools.sh). The jim_jobmanagers configuration procedure will be responsible for defining such entry points. The ability for a job to use such facilities (together with others, e.g. ups/upd) will be advertised to the broker; the presence of this attribute on the machine classads means that a job will find on its PATH the commands belonging to the advertised facility (e.g. sam, ups, etc.).
8. the product jim_sandbox contains the scripts that implement input sandboxing in JIM; the scripts will be principally used by the jim jobmanagers. We have to investigate whether we want to support the installation of this product by itself: the use case is if administrators want to enable the use of sandboxing for JIM from any of the globus jobmanagers by a vanilla job; the tailoring of this product will configure globus to treat the sandboxing scripts as a jobmanager tool (see point 7); the user executable will handle sandboxing using the product's scripts directly; this will probably happen transparently from the user by the use of thin wrappers (we still need to check whether the jobmanager makes available to the executable the use of its jobmanagers tools).

The xml database

Each product uses an xml document for its configuration. This document is the result of an interview with the local administrator/installer and it is driven from a product specific xml-meta-configurator template[1]. This configuration document is going to be available from the local xml db (its exact location in terms of the xml db name space is yet to be determined).

The name space of a product in the xml db

This paragraph discusses how to store configuration files in the XML database with consistent namespace conventions. It is a work in progress section.

[1] Defaults can be specified for some configuration values; for other "internal" values, the default should be written directly without asking the admin.

Namespace Hierarchy

All the information in the database is stored in a base collection /db. Collections form the namespaces for storing the configuration files related to every specific installation. Fully qualified host names forms the primary namespace, package name forms the secondary namespace and the package version forms the tertiary namespace. All the configuration files are stored as xml documents in the tertiary namespace.

Example:

Consider a site samadams.fnal.gov with following packages installed

- Samadams.fnal.gov
    1. sam_common v1_2
    2. sam_station_idl v2_3
    3. sam_common_idl v1_4
    4. corba_common v3_1
    5. jim_broker_client v1_2
    6. jim_advertise v2_4

    Suppose jim_advertise and jim_broker_client have configuration files as output of the xml_meta_configurator for the above installation then the database will have following entries for this site –

Collections	Documents
/db/samadams.fnal.gov/jim_advertise/v2_4	jim_advertise_config.xml
/db/samadams.fnal.gov/ jim_broker_client/v1_2	jim_broker_client_config.xml

Advantages

- Information is stored in a structured manner.
- Fixed set of rules to access the information.
- Ease of maintenance.
- Can store configuration for different versions of the same package installed on the system.

Independent of the database url used to store them. (local or maintained by Fermi lab)

API's/Tools for accessing configuration data from xml database

Package containing API's and tools to access the configuration data and xml database are listed below –

Package: Jim Config	
Dependencies: xml_common_lib, xmldb_client, xml_meta_configurator	
JimConfigManager	API's for managing the configuration information in xmldb

Package: Xmldb Client	
Dependencies:	
idmgr	API's to manage Job IDs
xmldb	API's to interface with the xmldb
xmldb_command	Command line interface for xmldb APIs
read_xml_value.py	Program to read any attribute from xmldb
set_xml_value.py	Program to set any attribute in xmldb

Package: Xml Common Lib	
Dependencies:	
PyXmlObject	API's to convert xml to python object
XmlDomUtils	API's for Dom